

Application of the ant colony algorithm for solving the job shop problem

Danil D Bukhovtsev

Siberian Federal University, Krasnoyarsk, Russian Federation

E-mail: wav9@yandex.ru

Abstract. The ant colony algorithm was first introduced by M. Dorigo, V. Maniezzo and A. Coloni in 1991. This algorithm is based on the behavior of a real ant colony when searching for food. The main area of application of the algorithm is the problem of combinatorial optimization. This paper describes the implementation of the algorithm, setting the parameters and considers its effectiveness in solving the job shop problem.

Keywords: ant colony algorithm, job shop problem, combinatorial optimization

Применение алгоритма муравьиной колонии для решения задачи календарного планирования

Данил Денисович Буховцев

Сибирский федеральный университет, Красноярск, Российская Федерация

E-mail: wav9@yandex.ru

Аннотация. Алгоритм муравьиной колонии впервые был представлен М. Дориго, В. Маньеццо и А. Колорни в 1991 году. Данный алгоритм основывается на поведении реальной колонии муравьев при поиске пищи. Основной областью применения алгоритма является задача комбинаторной оптимизации. В данной работе описываются реализация алгоритма, настройка параметров, а также рассматривается его эффективность при решении задачи календарного планирования.

Ключевые слова: алгоритм муравьиной колонии, задача календарного планирования, комбинаторная оптимизация

1. Введение

Многочисленные исследования колонии муравьев показывают, что ее поведение строго структурировано. Зная, что возможности отдельно взятого муравья ограничены (т.е. один муравей не способен напрямую общаться с другими собратьями), становится любопытно узнать, как же взаимодействуют муравьи, чтобы добиться такого организованного поведения всей колонии. Одним из способов взаимодействия является особое поведение муравьев в процессе поиска пропитания [1-4]. В процессе поиска каждая особь исследует окрестности вокруг муравейника, а затем возвращается в гнездо, чтобы сообщить об источнике пищи другим муравьям колонии. Для этого муравьи используют косвенный способ общения с помощью феромона, химического вещества, которое они могут откладывать для привлечения других муравьев. Каждый муравей, найдя источник пищи, откладывает часть феромона на обратном пути в гнездо, чтобы указать маршрут остальным. Накопленный феромон служит в качестве распределенной памяти, которую используют все остальные муравьи, а также помечает наиболее посещаемые маршруты между домом и источником пищи.

Муравьи, встречающие препятствие между муравейником и источником пищи, первоначально выбирают случайное направление движения. Как только на пути отложится часть феромона, это повлияет на выбор следующего муравья, столкнувшегося с тем же

препятствием, поскольку путь с большим количеством феромона с точки зрения вероятности имеет больший вес и привлекательность (рисунок 1).

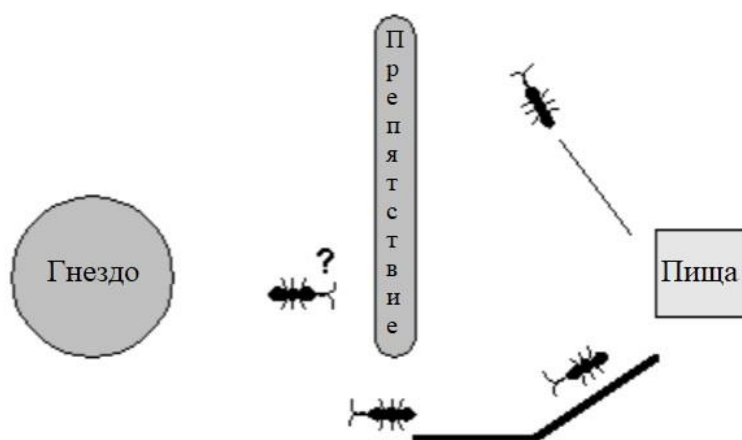


Рисунок 1. Положительная обратная связь с накопленным и испарившимся феромоном сходится к минимальному расстоянию пути.

Муравьи, перемещаясь по одному и тому же маршруту, помогают накапливать и поддерживать высокий уровень феромона, в то время как феромон, отложенный на малопроезжих тропах с большими длинами, начинает испаряться. Этот принцип уменьшает вероятность того, что последующие муравьи выберут самый длинный путь.

Вдохновленный этим поведением, М. Дориго разработал алгоритм, названный муравьиным алгоритмом, который использует популяцию агентов, взаимодействующих посредством распределённой памяти, и который может быть применен к комбинаторным задачам оптимизации [1-4]. В данной работе представлено применение алгоритма для решения задачи календарного планирования. Целью работы является получение оптимального набора входных параметров алгоритма, позволяющего получать решения хорошего качества. Содержание статьи выглядит следующим образом: во втором разделе представлено описание муравьиного алгоритма, в третьем разделе описывается задача календарного планирования, а в четвертом разделе рассматривается вопрос о том, как применить алгоритм для решения поставленной задачи. В пятом разделе описывается процесс поиска оптимального набора параметров. В шестом разделе представлены полученные с применением алгоритма результаты. И, наконец, в седьмом разделе сделаны выводы по работе.

2. Описание алгоритма

Основной принцип алгоритма заключается в наличии популяции l муравьев, которые на каждой итерации строят решения задачи комбинаторной оптимизации. Алгоритм представляет каждую задачу в виде связного графа, в котором муравьи перемещаются по ребрам от одной вершины к другой и таким образом строят маршруты, представляющие собой

решения. Каждый муравей выбирает следующую вершину для посещения в соответствии с правилом перехода:

$$p_{ij}(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot \left[\frac{1}{d_{ij}}\right]^\beta}{\sum_{j \in v} [\tau_{ij}(t)]^\alpha \cdot \left[\frac{1}{d_{ij}}\right]^\beta}, \quad (1)$$

Где τ_{ij} – количество феромона между вершиной i и вершиной j ;

d_{ij} – эвристическое расстояние между вершиной i и вершиной j ;

p_{ij} – вероятность перехода из вершины i в вершину j .

Параметры α и β регулируют «жадность» и «стадность» алгоритма соответственно. В зависимости от задачи набор возможных для посещения вершин из текущей может различаться. Если следующая для посещения вершина не может быть выбрана в связи с ограничениями задачи, то вероятность ее посещения считается равной нулю. В соответствии с этими принципами вершины с большим количеством феромона и находящиеся ближе к текущей вершине (с точки зрения эвристического расстояния) будут иметь более высокую вероятность посещения.

После того, как все муравьи построили решения, каждое из которых представляет собой последовательность посещенных вершин, цикл завершается и применяется правило обновления феромона. Правило глобального обновления феромона выглядит следующим образом:

$$\tau_{ij}(t+n) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}(t+n), \quad (2)$$

$$\Delta\tau_{ij}(t+n) = \begin{cases} \frac{Q}{f_{evaluation}(min)} \\ 0, \text{ в противном случае} \end{cases}, \quad (3)$$

Где ρ – коэффициент испарения;

Q – количество феромона на единицу расстояния.

Данное правило состоит из двух этапов. На первом этапе испаряется часть феромона на всех ребрах графа. На втором этапе происходит приращение феромона на ребрах, входящих в наикратчайший найденный маршрут на текущей итерации. Это приращение обратно пропорционально некоторой заданной оценочной функции. На следующих циклах вершины, принадлежащие лучшему найденному решению, будут иметь более высокую вероятность посещения, таким образом в алгоритме используется некий принцип обучения с подкреплением.

3. Проблема календарного планирования

В данной статье рассматривается стандартная модель задачи календарного планирования с n задачами и m станками, обозначаемой следующим образом:

$$n/m/G/C_{max} \quad (4)$$

Параметр G указывает, что задания связаны с технологическими правилами производства, описывающими порядок их обработки станками. Порядок обработки указывается в матрице T . Пример такой матрицы представлен ниже:

$$T = \begin{bmatrix} M1 & M2 & M3 \\ M2 & M3 & M1 \end{bmatrix} \quad (5)$$

Строка матрицы представляет собой задание, указывающее на последовательность обработки станками. Каждый элемент матрицы T называется операцией. Время обработки каждой операции задается матрицей P :

$$P = \begin{bmatrix} t(O_{11}) & \dots & t(O_{1m}) \\ \vdots & \ddots & \vdots \\ t(O_{n1}) & \dots & t(O_{nm}) \end{bmatrix} \quad (6)$$

Матрицы T и P вместе определяют проблему календарного планирования. Параметр C_{max} обозначает минимально возможное время обработки в цехе и используется для минимизации с помощью функции оценки. Значение C_{max} эквивалентно производственному времени, которое требуется для завершения всех работ с соблюдением наложенных ограничений на использование станков.

4. Применение алгоритма

На первом шаге необходимо представить входные данные в виде связного графа. Для этого вернемся к технологической матрице T , которая была рассмотрена в предыдущем разделе. В [1] была предложена идея представления данных в виде графа. На рисунке 2 приведен пример графа, построенный на основе модели $2/3/G/C_{max}$.

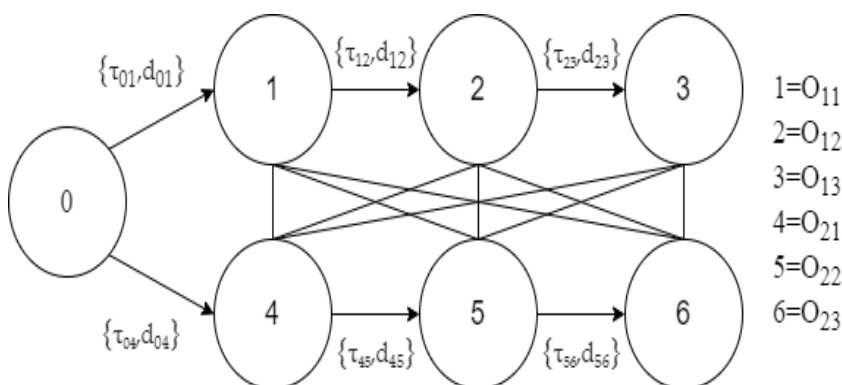


Рисунок 2. Представление модели $2/3/G/C_{max}$ в виде графа.

Вершины графа представляют операции, заданные матрицей T (например, O_{11} указывает на элемент T_{11} и обрабатывается станком $M1$). Операции, принадлежащие одному заданию, соединяются дугами с соблюдением порядка обработки. Остальные линии являются ребрами. Максимальное число вершин графа определяется по формуле:

$$V = n * m + 1 \quad (7)$$

Каждый муравей начинает построение маршрута из добавленной нулевой вершины. Количество ребер графа определяется по следующей формуле:

$$E = \frac{|O| \cdot (|O| - 1)}{2} + n, \quad (8)$$

Где $|O|$ – общее количество операций.

Каждое ребро связано с парой значений $\{\tau_{ij}, d_{ij}\}$, представляющие количество феромона и эвристическое расстояние между вершинами. В качестве эвристического расстояния между вершинами i и j в данной задаче можно использовать время обработки соответствующей операции, представленное в матрице P . Для данной эвристики важным замечанием является то, что в случае двунаправленных ребер их значения несимметричны. Это означает что расстояние от вершины i до вершины j не всегда равно расстоянию от вершины j до вершины i . Следовательно, количество феромона на двунаправленных ребрах также должно быть определено для обоих направлений. Пространственная сложность алгоритма определяется следующим образом:

$$SC = O([n * m][n * m]) \quad (9)$$

Кроме того, алгоритм должен гарантировать, что предлагаемые решения являются возможными. Для этого система должна проверять, что каждый муравей выбирает вершины для посещения в порядке, не нарушающем порядок обработки операций. Это реализовано с помощью идеи, описанной К. Бирвитом в [5], где предлагается использование списков. Таким образом, каждый муравей будет иметь 3 списка:

- G – список, который содержит еще не посещенные вершины;
- S – список, который содержит узлы, разрешенные для посещения в текущей итерации;
- $Tabu$ – список, содержащий уже посещенные вершины.

Первоначально все муравьи будут помещены в вершину с номером 0, и, если обратиться к примеру на рисунке 2, списки для k муравья будут выглядеть следующим образом:

$$G_k = \{O_{11}, O_{12}, O_{13}, O_{21}, O_{22}, O_{23}\}, \quad (10)$$

$$S_k = \{O_{11}, O_{21}\}, \quad (11)$$

$$Tabu_k = \{\} \quad (12)$$

После завершения цикла каждый муравей должен содержать в списке табу последовательность всех вершин, посещенных только один раз и в порядке, который определен порядком обработки операций. Временная сложность алгоритма определяется выражением:

$$TC = O(NC * l * [n * m]), \quad (13)$$

где NC – количество циклов;

l – количество муравьев.

5. Настройка параметров

Параметры алгоритма можно разделить на две независимые подгруппы: те, которые влияют на выбор вершины при переходе (α и β), и те, которые определяют обновление феромона (коэффициент испарения ρ и количество муравьев l). Параметры Q (количество феромона на единицу расстояния) и τ_0 не оказывают большого влияния на производительность алгоритма. Для начала необходимо найти такой набор параметров $\{\alpha, \beta\}$, при котором алгоритм сходится к заранее известному оптимуму выбранной модели. Для этого была выбрана модель $6/6/G/C_{max}$, предложенную Д. Мутом и Г. Томпсоном, взятую из [6], с оптимумом $C_{max} = 55$. Значение коэффициента испарения и количество муравьев оставались константой и равнялись $\rho = 0.01$ и $l = 36$ соответственно. Результаты запусков приведены в таблице 1.

Таблица 1. Количество циклов, необходимое для достижения оптимума при различных значениях α и β .

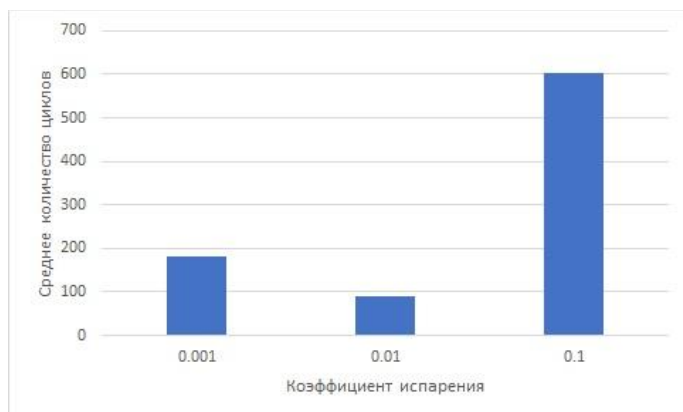
β	$\alpha = 0$	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$
0	>1000	>1000	>1000	>1000	>1000
1	>1000	>1000	>1000	>1000	>1000
2	>1000	>1000	>1000	>1000	>1000
3	>1000	>1000	>1000	>1000	>1000
4	>1000	614	>1000	150	>1000
5	>1000	326	442	249	>1000
6	>1000	437	314	202	228
7	>1000	283	247	148	267
8	>1000	310	260	170	121
9	>1000	293	187	164	96
10	>1000	186	203	132	96

В случае, когда $\alpha = 0$ и $\beta = 0$, каждый муравей выбирает вершину для перехода случайным образом, таким образом вероятность нахождения алгоритмом оптимума крайне мала. В случае, когда $\alpha = 0$ и β принимает любое значение, муравьиная колония не обращает внимания на уровень феромона, и каждый муравей выполняет эвристический жадный поиск по вероятности, таким образом алгоритму требуется больше тысячи итераций для нахождения оптимума. В случае, когда $\beta = 0$ и α принимает любое значение возникает аналогичная ситуация: муравей не принимает во внимание эвристическое расстояние, а сосредотачивается лишь на уровне феромона на ребрах. При $\beta = 10$ и увеличении значения параметра α количество итераций для нахождения оптимума уменьшается. Данная тенденция сохраняется и при $\alpha > 4$ (таблица 2).

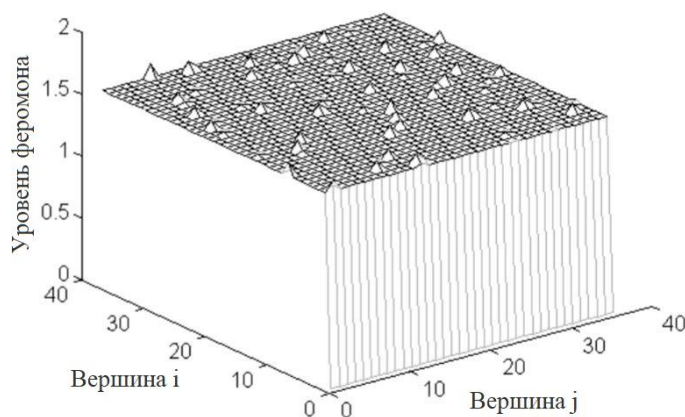
Таблица 2. Количество циклов, необходимое для достижения оптимума при различных значениях α и $\beta = 10$.

β	$\alpha = 5$	$\alpha = 6$	$\alpha = 7$	$\alpha = 8$	$\alpha = 9$	$\alpha = 10$
10	79	93	73	77	50	49

После нахождения оптимального набора параметров α и β можно исследовать влияние параметров, определяющих обновление феромона. На рисунке 3 показана работа алгоритма при различных значениях коэффициента испарения ρ .

**Рисунок 3.** Количество циклов, необходимое для достижения оптимума при различных значениях ρ .

В случае, когда $\rho = 0.1$, количество циклов, требуемое для нахождения оптимума, максимально. Это связано с тем, что при слишком высокой скорости испарения только ребра, входящие в маршрут с наименьшей длиной, получают обновление феромона каждый цикл. У остальных ребер феромон быстро уменьшается до нуля с увеличением числа циклов из-за испарения. С каждым новым циклом этот эффект усиливается, и муравьи будут уделять большее внимание ребрам, получающим обновление феромона, ища в небольшой области локального минимума. Рисунки 4-5 демонстрируют уровень феромона после прохождения 25 циклов при оптимальном и худшем значениях коэффициента ρ .

**Рисунок 4.** Уровень феромона после 25 циклов при $\rho = 0.01$.

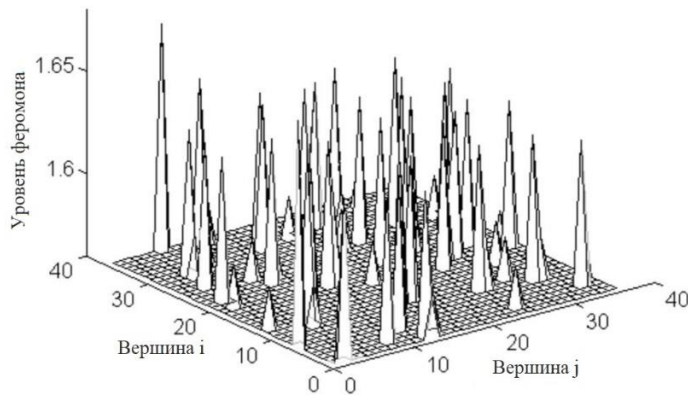


Рисунок 5. Уровень феромона после 25 циклов при $\rho = 0.1$.

Влияние количества муравьев в колонии на производительность алгоритма, представлено на рисунке 6.

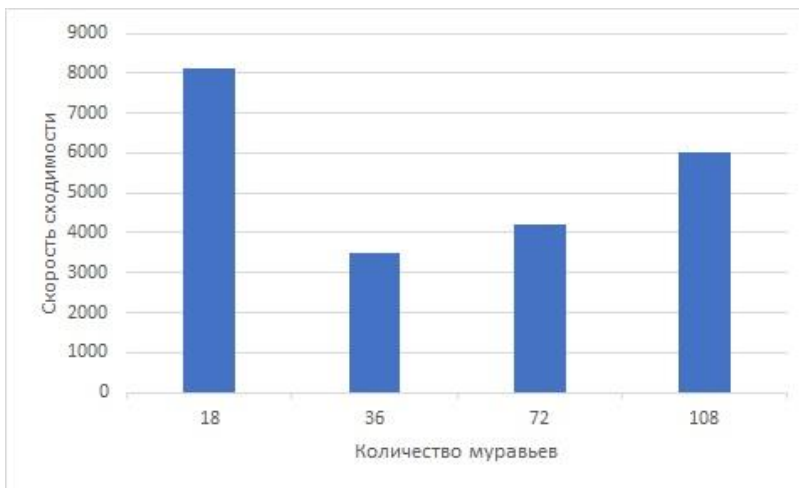


Рисунок 6. Влияния количества муравьев на скорость сходимости.

Скорость сходимости алгоритма рассчитывается, как среднее количество циклов для достижения оптимума, умноженное на количество муравьев. Из рисунка 6 следует, что оптимальное количество муравьев должно приблизительно равняться количеству вершин в графе.

6. Результаты

В данном разделе продемонстрированы результаты работы алгоритма на более сложных тестовых задачах с набором входных параметров, взятых из раздела 5. Однако, в начале рассмотрим результаты прогонов на вышеупомянутой модели $6/6/G/C_{max}$.

На рисунке 7 приведено сравнение функции оценки при различных значениях параметров.

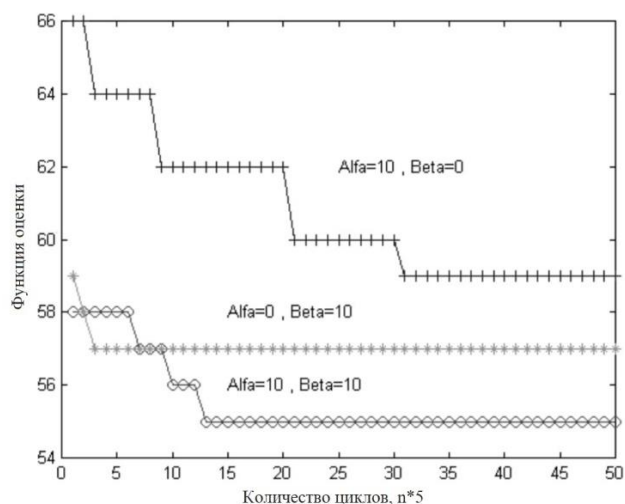


Рисунок 7. Значения функции оценки при различных входных параметрах.

Изучив полученные значения функции оценки, можно сделать вывод, что поиск оптимума оказывается в ловушке в малой окрестности локального минимума. Чтобы этого не происходило, необходимо ввести новый оператор, называемый вариацией, который при выполнении определенного условия, заменяет правило перехода:

$$\left\{ \begin{array}{l} (1), \text{ если } n \geq v, \\ p_{ij} = \frac{\frac{1}{d_{ij}}}{\sum_{j \in v} \frac{1}{d_{ij}}}, \text{ в противном случае,} \end{array} \right. \quad (14)$$

где n – случайное число из отрезка $[0..1]$;

v – случайное число из отрезка $[0..1]$.

Влияние различных значений вариации приведено на рисунке 8.

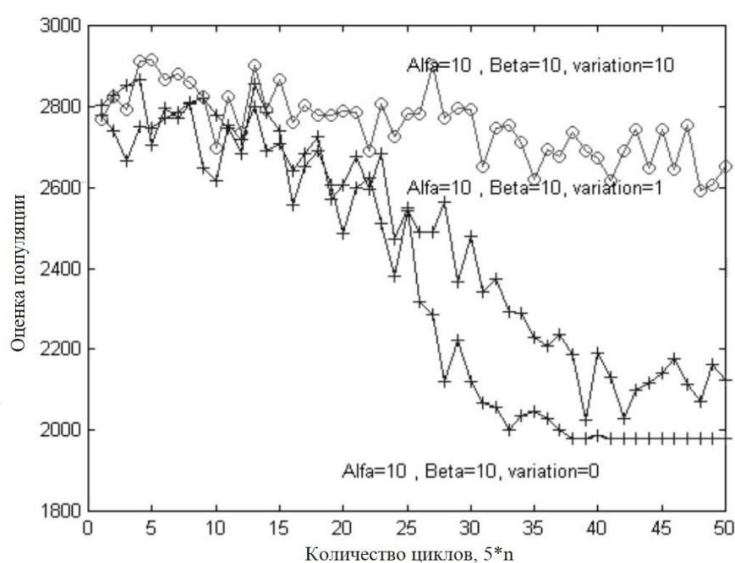


Рисунок 8. Влияние оператора вариации положительно сказывается на результатах поиска.

Без оператора вариации функция оценки популяции сходится к постоянному значению, что означает, что все муравьи выбирают один и тот же маршрут, что, конечно же, не может нас устраивать. Введение данного оператора можно рассматривать как некоторое насыщение восприятия феромонов каждым муравьем. При заданном проценте вариации за цикл муравьи будут нечувствительны к восприятию феромонов и, таким образом, будут вести свой поиск только на основе эвристической информации.

Алгоритм был протестирован на двух известных задачах, представленных в [6]. Первая модель $10/10/G/C_{max}$ была представлена вышеупомянутыми деятелями Д. Мутон и Г. Томпсоном с оптимумом $C_{max} = 930$. Вторая модель была $20/10/G/C_{max}$ представлена С. Лоуренсом с оптимумом $C_{max} = 1218$. Полученные с помощью алгоритма результаты для различных значений параметров e и ν представлены в таблице 3.

Таблица 3. Результаты работы алгоритма при $\alpha = 10$ и $\beta = 10$.

Модель	C_{max}	e	ν	% в пределах оптимума
$10/10/G/C_{max}$	1052	1	0	13
$10/10/G/C_{max}$	1006	1	1	8
$10/10/G/C_{max}$	1063	1	10	14
$10/10/G/C_{max}$	1019	2	2	9.5
$10/10/G/C_{max}$	1054	3	1	13.3
$10/10/G/C_{max}$	1041	4	3	12
$20/10/G/C_{max}$	1604	1	0	31
$20/10/G/C_{max}$	1607	1	1	31.9
$20/10/G/C_{max}$	1535	3	1	26

7. Заключение

В данной работе был продемонстрирован способ решения проблемы календарного планирования с применением муравьиного алгоритма. Цель работы заключалась в том, чтобы исследовать поведение алгоритма и качество полученных решений при различных значениях входных параметров. Параметры α и β , как уже было сказано ранее, влияют на скорость сходимости алгоритма и конечный результат работы. Чтобы алгоритм сходился к удовлетворительному решению очень важно подобрать оптимальное значение коэффициента испарения ρ . Это направит поиск в предпочтительные области в пространстве поиска и уменьшит вероятность попадания в ловушку локального минимума. Введение параметра вариации ν (аналогично оператору мутации в генетических алгоритмах) позволяет исследовать субоптимальные области пространства поиска, не теряя при этом способности алгоритма выходить из тупиков. Муравьиный алгоритм был протестирован на некоторых

более сложных тестовых задачах. Для модели $10/10/G/C_{max}$ было зафиксировано отклонение от оптимума на 8%, а для модели $20/10/G/C_{max}$ – на 26%.

Основное преимущество данного алгоритма заключается в том, что он хорошо справляется с задачами комбинаторной оптимизации, представленными в виде несимметричного связного графа. Единственной доработкой, которую необходимо было ввести, являлось расширение таблицы феромонов. Это увеличивает его пространственную сложность, но не требует дополнительной вычислительной мощности. С другой стороны, именно временная сложность является основным недостатком муравьиного алгоритма. Временная сложность алгоритма по сравнению с генетическими алгоритмами экспоненциально возрастает с увеличением размера популяции. Это связано с тем, что на каждом цикле муравьиного алгоритма необходимо строить решения для всех муравьев. Именно поэтому следует применять генетический алгоритм к задачам комбинаторной оптимизации, которые могут быть представлены в виде симметричного графа.

Список литературы

- [1] Dorigo, M The Ant System: Optimization by a colony of cooperating agents / M. Dorigo, V. Maniezzo, A. Coloni // IEEE Trans. Syst, Man, Cybern. – 1996. – № 26(2). – P. 29-41.
- [2] Dorigo, M. Ant colony System: A Cooperative Learning Approach to the Travelling Salesman Problem / M. Dorigo, L. M. Gambardella // IEEE Trans. On Evolutionary Computation. – 1997. – № 1(1).
- [3] Dorigo, M. Distributed Optimization by Ant Colonies / M. Dorigo, V. Maniezzo, A. Coloni // Proceedings of ECAL91 – European Conference on Artificial Life, Elsevier Publishing. – 1991. – P. 134-142.
- [4] Dorigo, M. An Investigation of some properties of an Ant Algorithm / M. Dorigo, V. Maniezzo, A. Coloni // Proceedings of the Parallel Problem Solving from Nature Conference (PPSN92), Brussels, Belgium, Elsevier Publishing. – 1992. – P 509-520.
- [5] Bierwith, C. A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms / C. Bierwith // Department of Economics, University of Bremen, Germany. – 1995. – № 17. – P. 87-92.
- [6] Job Shop Scheduling Benchmark, OR-library. <http://mscmga.ms.ic.ac.uk/jeb/orlib/jobshopinfo.html>.